Thesis for The Degree of Doctor of Philosophy

A Process-oriented Approach for Migrating Software to Heterogeneous Platforms

Hugo Sica de Andrade

Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden, 2021

A Process-oriented Approach for Migrating Software to Heterogeneous Platforms

Hugo Sica de Andrade

ISBN 978-91-7905-439-7
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 4906.
ISSN 0346-718X

Technical Report No 193D
Department of Computer Science & Engineering
Division of Software Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2021.

To my father Hélio, my mother Gina, and my sister Débora.

Abstract

Context: Heterogeneous computing, i.e., computing performed on processors of different types—such as combination of CPUs and GPUs, or CPUs and FPGAs—has shown to be a feasible path towards higher performance and less energy consumption. However, this approach imposes a number of challenges on the software side that must be addressed in order to achieve the aforementioned advantages.
Objective: The objective of this thesis is to improve the process of software deployment on heterogeneous platforms. Through a detailed analysis of the state-of-the-art and state-of-the-practice, we aim to provide a reasoning framework for engineers to migrate software to be executed on such platforms.
Method: To achieve our goal, we conducted: (i) a literature review in the form of a systematic mapping study on software deployment on heterogeneous platforms; (ii) a multiple case study in industry that highlights the main challenges and concerns in the state-of-the-practice in the area; and (iii) a study in which we propose and evaluate a decision framework to guide engineers in migrating software for execution on heterogeneous platforms, with a case study in the automotive domain.
Results: In the mapping study, we provided a thorough classification of the identified concerns and approaches to deploying software on heterogeneous platforms. Among other findings, we discovered a lack of holistic approaches that include development processes, as well as few validation studies in industrial contexts. In the second study, we discovered and analyzed common practices and challenges that companies face when using heterogeneous platforms. One of such challenges is related to the lack of approaches that cover the software development lifecycle. In the third study, we proposed a decision framework that guides engineers in the process of reasoning for migrating software for execution on heterogeneous platforms. It consists of five stages (*assessing*, *re-architecting*, *developing*, *deploying*, *evaluating*), each containing a set of aspects to be addressed through the answers to predefined questions.
Conclusions: This thesis addresses a gap that was identified in both theory and practice concerning the lack of holistic approaches to migrate software for execution on heterogeneous platforms. Our proposed approach addresses the problem through systematic guidance for engineers.
Future work: In the future, we intend to further refine the proposed framework through case studies in domains other than automotive. We will explore its integration with existing software engineering processes in industrial contexts, performing in-depth analysis of the required adaptations and providing detailed solutions within the stages of the framework.

Keywords

Software Engineering, Software Deployment, Software Architecture, Heterogeneous Platforms

Acknowledgements

This journey has been possible with the support of the following individuals.

Ivica Crnkovic: thank you for your mentoring during the last several years. Words cannot properly define my gratitude for your endurance, patience and support when I had my ups and downs. Learning has been my main goal all along, and I have definitely learned a lot from you.

Jan Bosch and Christian Berger: thank you for bringing so much perspective to my life inside and outside work. Through completely different strategies, you constantly raised the bar and made me believe I could succeed.

Federico Giaimo: thank you for being such an important piece in my PhD puzzle. More than an office mate, you have been the person who provided me with support when times were tough and laughter even in the smallest things. One of the most valuable outcomes of this experience is our friendship.

Truong Ho-Quang, Hang Yin, Yue Kang, Ricardo Caldas: thank you for the everyday exchanges we had sharing the office. It was great to have someone to talk to when I needed a break.

Jan, Rodi, Abdullah, Lucas, Alessia, Gul, Michel, David, Miroslaw, Patrizio, Aiswarya, Lucy, Hamza, Salome, Ann, Magnus, Terese, Sergio, Grischa, Rebekka, Katja, Pier, Mukelabai, Vard, Ivan, Juraj, Zdravko: thank you for being great colleagues in this journey. Even in our smallest interactions, such as corridor chats, you made a difference in my life.

Hélio, Gina and Débora: thank you for your unconditional support throughout my entire life, no matter what I chose of it. I would not be able to make it without you. Obrigado! Amo vocês.

Anna Pennlund: thank you for all your kind gestures and daily doses of support. There were times where you believed in me more than I did myself. I will never be able to fully express my gratitude to you. You have been the best partner I could have asked for. Älskar dig!

List of Publications

Included publications

The following publications are included in this thesis:

[A] H. Andrade, J. Schröder, I. Crnkovic "Software Deployment on Heterogeneous Platforms: A Systematic Mapping Study"
IEEE Transactions on Software Engineering journal (TSE), 2019. [also presented as Journal First article at the International Conference on Software Engineering (ICSE), 2020].

[B] H. Andrade, I. Crnkovic "A Review on Software Architectures for Heterogeneous Platforms"
Proceedings of the 25th Asia-Pacific Software Engineering Conference (APSEC). Nara, Japan, 4-7 December, 2018.

[C] H. Andrade, L. Lwakatare, I. Crnkovic, J. Bosch "Software Challenges in Heterogeneous Computing: A Multiple Case Study in Industry"
Proceedings of the 45th IEEE Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA). Kallithea-Chalkidiki, Greece, 28-30 August, 2019.

[D] H. Andrade, C. Berger, I. Crnkovic, J. Bosch "Principles for Re-architecting Software for Heterogeneous Platforms"
Proceedings of the 27th Asia-Pacific Software Engineering Conference (APSEC). Singapore, 1-4 December, 2020.

[E] H. Andrade, O. Benderius, C. Berger, I. Crnkovic, J. Bosch "HPM-Frame: A Decision Framework for Executing Software on Heterogeneous Platforms"
Journal of Systems and Software (JSS), 2020 [Submitted].

Other publications

The following papers were also published during my studies, although not included in this thesis:

[a] H. Andrade, I. Crnkovic, J. Bosch "Refactoring Software in the Automotive Domain for Execution on Heterogeneous Platforms"
IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). Virtual event, 13-17 July, 2020.

[b] F. Giaimo, H. Andrade, C. Berger "Continuous Experimentation and the Cyber-Physical System challenge. An overview on literature and the industrial perspective"
Journal of Systems and Software (JSS), 2020.

[c] F. Giaimo, H. Andrade, C. Berger "The Automotive Take on Continuous Experimentation: A Multiple Case Study"
Proceedings of the 45th IEEE Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA). Kallithea-Chalkidiki, Greece, 28-30 August, 2019.

[d] A. Rodrigues, G. Rodrigues, A. Knauss, R. Ali, H. Andrade "Enhancing Context Specifications for Dependable Adaptive Systems: A Data Mining Approach"
Information and Software Technology journal (IST), 2019.

[e] G. N. Rodrigues, A. Knauss, F. Guimaraes, G. Rodrigues, H. Andrade, J. Araujo, R. Ali "GoalD: A Goal-Driven Deployment Framework for Dynamic and Heterogeneous Computing Environments"
Information and Software Technology journal (IST), 2019.

[f] D. Feitosa, A. Ampatzoglou, P. Avgeriou, F. J. Affonso, H. Andrade, K. R. Felizardo, E. Y. Nakagawa "Design Approaches for Critical Embedded Systems: A Systematic Mapping Study"
In: Damiani E., Spanoudakis G., Maciaszek L. (eds) Evaluation of Novel Approaches to Software Engineering. ENASE 2017. Communications in Computer and Information Science, vol 866. Springer, Cham, 2018.

[g] P. Masek, M. Thulin, H. Andrade, C. Berger, O. Benderius "Systematic Evaluation of Sandboxed Software Deployment for Real-time Software on the Example of a Self-Driving Heavy Vehicle"
Proceedings of the 19th IEEE International Conference on Intelligent Transportation Systems (ITSC). Rio de Janeiro, Brazil, 1-4 November, 2016.

[h] H. Andrade "Investigating Software Deployment on Heterogeneous Platforms"
Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). Venice, Italy, 5-8 April, 2016.

[i] H. Yin, F. Giaimo, H. Andrade, C. Berger, I. Crnkovic "Adaptive Message Restructuring using Model-Driven Engineering"
In: PLatifi S. (eds) Information Technology: New Generations. Advances in Intelligent Systems and Computing, vol 448. Springer, Cham, 2016.

[j] H. Andrade, F. Giaimo, C. Berger, I. Crnkovic "Systematic Evaluation of Three Data Marshalling Approaches for Distributed Software Systems"
Proceedings of the Workshop on Domain-Specific Modeling (DSM @ SPLASH). Pittsburgh, USA, 27 October, 2015.

[k] F. Giaimo, H. Andrade, C. Berger, I. Crnkovic "Improving Bandwidth Efficiency with Self-Adaptation for Data Marshalling on the Example of a Self-Driving Miniature Car"
Proceedings of the 9th European Conference on Software Architecture Workshops (ECSAW). Dubrovnik/Cavtat, Croatia, 7-11 September, 2015.

[l] H. Andrade, E. Almeida, I. Crnkovic "Architectural Bad Smells in Software Product Lines: An Exploratory Study"
Proceedings of the Working IEEE/IFIP Conference on Software Architecture Companion Volume (WICSA). Sydney, Australia, 7 April, 2014.

Personal Contribution

I was the main driver of all studies appended in this thesis.

In Papers A and B, I have selected the most appropriate method of literature review and written the review protocol, which included a clear motivation for the study, research goal and questions, the inclusion/exclusion criteria for selecting primary studies, and possible classification attributes (facets). I led discussions with the coauthors throughout the whole process. I conducted pilot studies by testing different search strings into the search engines of selected digital libraries. I conducted the paper selection process with the help of one researcher by applying the predefined criteria, with focus on the RQs. I read and extracted relevant data from the papers, managing information through spreadsheets, notes and reference managers. I qualitatively and quantitatively analyzed the data in the search for clusters, trends and gaps, presenting the findings in the form of charts, tables, graphs and *mindmaps*. I wrote most of the papers.

In Paper C, I structured the idea of the paper based on the findings from our previous studies. I held initial discussions with the coauthors to align concepts, ideas and goals for the study. I conducted face-to-face semi-structured interviews with the participating experts based on a predefined set of questions and topics of interest. The interview guide was created by me and evolved through discussions with the coauthors. I collected the data through notes and recordings of the interviews with the help of one researcher, which were later transcribed. I thoroughly analyzed the data in the search for patterns, resulting in *mindmaps* for structure. The topics contained brief rationales, which were later transformed into the first sketches of the model presented in the paper. I created both the model and the summaries, evolving them through discussions with the coauthors. I wrote most of the paper.

In Papers D and E, I elaborated the research plan based on our previous studies and through discussions with the coauthors. I reviewed the studies and defined the research design based on the defined research goal. I sketched the first versions of the proposed framework and presented them in multiple occasions, such as research workshops, face-to-face meetings and video calls. I created a questionnaire with both general and specific questions about the proposed framework. I refined the framework by creating different stages, aspects and questions, along with its interaction with a typical software engineering process. The impression of the experts was captured through both their answers to the questionnaire and discussions during workshops, resulting in improvements of the framework. To evaluate it, I elaborated the design and conducted a case study with the help of two researchers in the process of extracting the software architecture and implementing the case application. I applied the proposed framework to the case, discussing and presenting the results in the form of text, tables and figures. I wrote most of the papers.

Contents

Chapter 1

Introduction

The role of software continues to gain importance in modern society. Computer programs now have a long track of strongly pushing for evolution in domains far beyond the obvious, such as applied mathematics. Software has completely transformed the lives of billions of people in their common activities through ubiquitous computing. Industry has also greatly benefited from extensive digitalization of processes and automatization of activities previously performed by humans. There is currently no forecast that is contrary to the idea that software will continue to gain space in society. The most prominent use of software nowadays is arguably in the field of Artificial Intelligence (AI), which focuses on providing solutions to very complex computational problems. Machine learning algorithms, for instance, allow for automatic data analysis through which systems can *learn* from data, identify patterns and make decisions with minimum human intervention [1].

One particular aspect becomes relevant as more and more responsibilities are attributed to software and realized through the implementation of functionalities: *performance*. The performance of a system also relies on its hardware capability, which must not only be sufficient for a smooth execution of a predetermined set of features, but also cheap enough to make products financially viable. This is particularly challenging in the case of embedded systems, which are often limited in resources and many times have real-time and interface constraints [2]. In the past, the requirements for hardware performance were fulfilled by (i) boosting the frequency of processing units (PUs) or by (ii) adding transistors onto processors. Since boosting the frequency of processors is becoming difficult [3], performance is primarily achieved by increasing the transistor count. However, the number of transistors built on chips has broken several records in recent times (e.g., Cerebras Wafer Scale Engine featuring over 1.2 *trillion* transistors [4]). Making use of this many transistors is difficult due to the inherent complexity in managing the circuit logic.

The most promising way to fulfill these high demands for performance is to employ heterogeneous platforms, i.e., hardware platforms consisting of more than one type of processors. Heterogeneous platforms combine processors such as: multi-core Central Processing Units (CPUs), Graphics Processing Units (GPUs) Application-Specific Integrated Circuits (ASICs) and

Field-Programmable Gate Arrays (FPGAs). Using such platforms creates the impression of dedicated units that can be adapted to a wide range of application domains. These dedicated units can significantly increase the overall system's performance and energy management through, for instance, optimizing the workload distribution according to the types of data to be processed. There currently exists a variety of processing units that can be used as accelerators, though the development of technologies has made GPUs and FPGAs popular due to their flexibility in improving the performance of applications in several domains. In particular, GPUs have gained significant attention due to their cost benefit in processing graphics in both industrial and consumer electronics contexts. In the latter, the popularization is mainly driven by personal computers and mobile phones.

Despite the fact that the difficulty in programming for such platforms is decreasing [5], there are multiple concerns that must be addressed in order to handle the inherent complexity of both hardware and software in such environments. Compared to traditional CPU-based programming, software development for heterogeneous platforms requires the developer to acknowledge several aspects of the underlying hardware as soon as in the design phase, when different software deployment strategies may be modeled and analyzed. For instance, the memory shared between the processors must be explicitly managed even when using comprehensive hardware manufacturer development frameworks such as CUDA for NVIDIA GPUs [6]. Further, by using heterogeneous platforms, the software architecture is assumed to follow a primary/secondary pattern, as the main flow of the application is typically executed by the CPU while the computationally heavy portion of the algorithm is dispatched to the accelerator.

In real-world software engineering contexts, the adoption of heterogeneous platforms is gradual, driven by necessity, and very dependent on the application that is to be executed. It is common practice that software is first developed for execution on CPUs and then *migrated* to a solution based on multiple processors, due to prototyping and the evaluation of need for additional performance. The change is typically guided by a particular performance benchmark that is previously defined in the requirements and achieved through the selection of a particular hardware solution. Such software migration may require changes not only to the source code itself, but also to the overall software architecture, the development processes, and the business practices that are currently in place. In some cases, a software project might start already with heterogeneous platforms in place for execution of applications that are known to be highly demanding on computational power. However, even in such cases, the hardware costs must be justified through the performance benchmarking of implementations based on CPUs. In this sense, the migration of software for execution on heterogeneous platforms represents a relevant activity that must be addressed by engineers in the process of adopting heterogeneous platforms.

This thesis consists of an introduction chapter that covers the overall description of the research, followed by a collection of five papers. The papers have been formatted to fit the layout of the thesis, being included as individual chapters. This introduction chapter covers the overall description of the research and is organized as follows. Section 1.1 introduces the background of this work. Section 1.2 presents the research goal, scope, and research questions of this

thesis. Section 1.3, describes the research methodologies used in this work. Section 1.4, summarizes the contributions of this thesis and discusses their connections to the research questions. Section 1.5 presents an overview of the publications included in this thesis. Section 1.6 discusses the threats to the validity of this work. Then, Section 1.7 presents the conclusions. Finally, Section 1.8 describes our intentions for future work.

1.1 Background

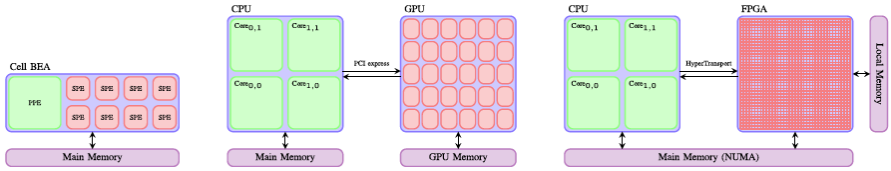
In this section, we provide the background for the main topics covered in this thesis: heterogeneous platforms (heterogeneous computing), software deployment, and software refactoring.

1.1.1 Heterogeneous platforms

During our investigation, we discovered that the literature refers to the term "*heterogeneous platform*" in different ways. Besides meaning different processors, we found that this term also refers to platforms containing processors of the same type, but with different capacities. For instance, a system that includes 2 CPUs with a different number of cores and/or clock frequencies is often called *heterogeneous*. Another situation in which the term is commonly found is when the types and other characteristics of the processors are omitted, being discussed only their differences in capacity. For example, strictly combinatorial problems consider a cost formula and a few performance attributes of the processors in order to determine the best deployment strategy.

In this thesis, we adopted the following definition of heterogeneous computing: *complex systems composed of different kinds of processing units which use different processing paradigms and are designed for different types of tasks which work together in order to provide the best processing performance for diverse computing needs* [7]. In this sense, we consider a heterogeneous platform as a hardware set consisting of at least two different types of processors that are specialized in different types of tasks.

An example of heterogeneous hardware architectures is shown in Figure 1.1 [3]. In Figure 1.1(a), the single-chip Cell Broadband Engine Architecture (CBEA) is depicted consisting of a traditional CPU core and eight single-instruction multiple data (SIMD) accelerator cores. Each core can run separate programs and communicate through a fast on-chip bus. Its main design criteria is to maximize performance while consuming minimum power. Figure 1.1(b) illustrates a GPU with 30 highly multi-threaded SIMD accelerator cores in combination with a standard multicore CPU. The GPU has superior bandwidth and computational performance. It is designed for high-performance graphics, where throughput of data is key. In Figure 1.1(c), a standard multi-core CPU is paired with an FPGA consisting of an array of logic blocks. FPGAs can also incorporate regular CPU cores on-chip, making it a heterogeneous chip by itself. FPGAs offer fully deterministic performance and are designed for high throughput, for example, in telecommunication applications.



(a) CBEA (b) CPU in combination with GPU (c) CPU in combination with FPGA

Figure 1.1: Example of heterogeneous hardware architectures as shown in [3]: (a) Cell Broadband Engine (heterogeneous chip), (b) a CPU in combination with a GPU, and (c) a CPU in combination with an FPGA.

1.1.2 Software deployment

The concept of *deployment* also varies according to the context in which the study is performed. For business research, it may refer to strategies for update releases of a mobile app. For technology research, deployment may refer to the tools that are used to facilitate and enable deployment, e.g., Docker [8]. For fundamental research, it may refer to the mathematical strategies to optimize load balance in a heterogeneous environment.

In the context of software engineering, *software deployment* comprises a set of activities resulting in a system that is available for use [9]. These activities can be very diverse and include a wide range of processes, such as users training, integration of new features into the existing system, the actual installation of software on the underlying hardware, etc. A typical deployment activity refers to the activities in the process of installing software on hardware, including the decision about the units in which software components will be executed (component allocation). The activities include partitioning the software system into components and planning their execution on different processing units.

As we conducted this work, we realized that the activities performed in the typical deployment stage are heavily influenced by activities in previous stages in the software process. For instance, we learned that one common way to realize deployment onto heterogeneous platforms is by using a development framework, which needs to be applied as soon as in the architecture phase. For this reason, we extend the concept of deployment to include all activities that are relevant throughout the software engineering process to successfully execute software onto a heterogeneous platform. Further, we focus on deployment from the software perspective rather than from the hardware perspective.

One example representation in the domain of cyber-physical systems is shown in Figure 1.2, where a deployment scenario is depicted. On the hardware side, there is a heterogeneous platform consisting of an FPGA, n CPUs and m GPUs that are available for processing data, and these units have interfaces with different types of sensors and actuators. The software is decomposed into components that can be deployed according to different configurations, while the following assumptions might be relevant: (i) instructions might execute in a shorter time on the FPGA when compared to CPUs or GPUs, however programming for FPGAs is complex and more time consuming; (ii) two dependent applications might be executed faster in different executing units, however the communication between them might be compromised by

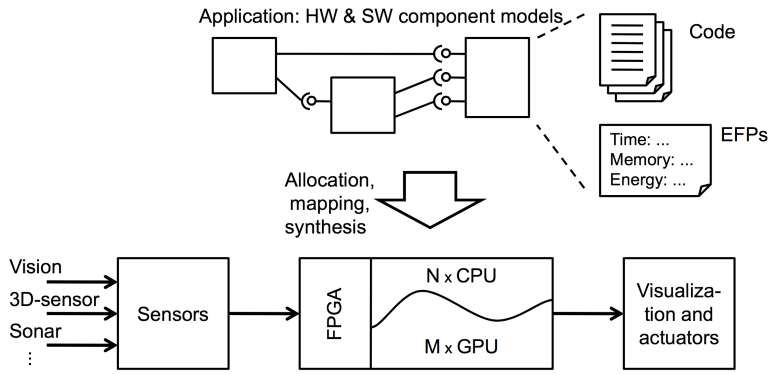


Figure 1.2: Generic modelling scenario of cyber-physical systems, depicting software deployment on a heterogeneous platform [10].

the available bandwidth; (iii) allocating components and highly parallelizable applications on a single executing unit (e.g., GPU) might be less complex, but also compromises energy efficiency. Other aspects may also be considered, such as the impact of the technology used to encapsulate processes for software deployment or the underlying environment in which the application is executed.

1.1.3 Software migration & decision process

The most important challenge to software engineering when employing heterogeneous platforms is the fact that legacy CPU-based applications are *not* ready for execution on such hardware. Due to the inherent differences in the architecture and development processes, the software must be *migrated* prior to deployment. The term refers to the set of steps that engineers take in order to make an existing CPU-based application executable on a heterogeneous platform. Migration is not only performed in legacy CPU-based applications, but also in the implementation of new functionalities, since a CPU-based implementation is typically used to measure performance of the system using accelerators. It is generally simpler and faster to create prototypes and proofs of concept, targeting execution on CPUs. In this case, the CPU-based implementation is used as benchmark to assess the behavior of applications deployed on heterogeneous platforms.

Software migration requires careful analysis of the existing requirements and functionalities in order to determine the extent to which they are relevant to the business. The process can be seen as an opportunity to improve the software itself as well as the development processes that are currently in use. In the case of legacy applications, it is not uncommon that the knowledge about technical details of the system is scarce across the organization, demanding that the business goals are revisited. Architecture erosion, i.e., a mismatch between the planned architecture and the actual architecture [11], is a phenomenon that might be identified and addressed in this process. Furthermore, the architectural pattern might be updated, such as in the case of a transformation from monolithic to a microservices architecture [12]. As there is no standard heterogeneous architecture currently in use, the cost of each migration seems

to include many architecture-specific issues that must be addressed. Most concerns in migrating are related to software/hardware mapping and the required changes to the interfaces, along with adaptations in the messages that are passed between components.

Among other tasks, the migration of applications to heterogeneous platforms includes *software refactoring* [13]. The term refers to the process of restructuring code without changing the application's external behavior while improving non-functional properties. The refactoring approach has become more prominent in recent years due to the rise in the number of programming models, languages and frameworks for expressing parallelism [14]. Furthermore, models that are specific to particular accelerators, such as GPUs, have also appeared and need to be considered [15]. Typically, parallelism must be expressed in initially sequential applications in order to fully take advantage of the capabilities of heterogeneous platforms. In terms of implementation details, the current technologies require programmers to explicitly handle memory spaces between different processing units. Thus, the process of migrating software must account for the increased complexity in the code in addition to the changes in the architecture. Furthermore, the migration might require changes in the existing software engineering processes that are in place, as well as in the adoption of new software development frameworks, tools, and deployment routines.

The aforementioned challenges to software migration must be addressed through a *decision process* in order to achieve the desired benefits of heterogeneous platforms. Particularly in industry, such process must be clear and systematic in guiding engineers due to the need of reproducibility. A typical decision process includes the identification of a decision, information gathering, and assessing alternative solutions [16]. In the context of heterogeneous platforms, an effective decision process must contain all relevant aspects of migration for engineers to reason upon and ultimately address according to their software engineering contexts. In this sense, the application of a step-by-step decision process based on predefined steps reduces the risks in the project connected to diverse interpretations and engineering actions.

1.2 Research Goal & Scope

The ultimate goal of this thesis is *to provide a decision framework for engineers in the process of migrating software for execution on heterogeneous platforms.* This goal implies an investigation of both academic and industrial contexts in order to provide: (i) an overview of the state-of-the-art of software deployment on heterogeneous platforms; (ii) an overview of the common challenges and practices in industry when developing software for heterogeneous platforms; and (iii) a decision framework that supports reasoning in the task of migrating software for execution on heterogeneous platforms.

The topic of heterogeneous platforms is often referred to as a discipline in the area of hardware engineering. In this work, however, we consider the software engineering perspective, which covers methods, processes, and techniques that enable and lead to the execution of software on such platforms. Based on the research goal, we formulated three research questions, as follows.

RQ1: What are the main concerns and approaches in software deployment on heterogeneous platforms?

As the first step towards understanding the area of research, we aimed to investigate the state-of-the-art of software deployment on heterogeneous platforms, focusing on the main concerns and approaches that can be found in the literature. The purpose of RQ1 is to obtain an overview of the body of knowledge in the area by revealing concerns and approaches that are relevant when deploying software on heterogeneous platforms.

RQ2: What are the challenges in industry concerning software development for heterogeneous platforms?

The purpose of RQ2 is to obtain an overview of the scenario in industry regarding the development of software for heterogeneous platforms. For this, we aimed at identifying the challenges that must be addressed and characteristics that are relevant about companies in different domains when using heterogeneous platforms.

RQ3: What is a relevant process for engineers to follow when migrating software to heterogeneous platforms?

Based on the overviews obtained from both academia and industry contexts, we narrow the scope of research in order to propose an approach addressing the migration problem. With RQ3, we aim to identify aspects that are relevant to successfully migrate software for execution on heterogeneous platforms.

1.3 Research Methodology

We used four different research methods to address the formulated research questions, as follows.

1.3.1 Systematic literature reviews

To address RQ1, we conducted a systematic literature review in the form of a mapping study. Mapping studies differ from classic systematic literature reviews in their broadness and depth [17, 18]. Instead of rigorously searching, analyzing and assessing studies, selected information is extracted from the primary studies in order to obtain an overview of the current state-of-the-art of research in a particular field.

We aimed at performing a systematic approach to increase reliability of the study and enable reproducibility in the future. The search included the most relevant academic databases in computer science and followed a set of predefined inclusion and exclusion criteria. After the selection of studies from the libraries, we performed the snowballing procedure [19] to also cover related papers. The review selected 146 primary studies, and therefore collected a large amount of data to be analyzed and discussed. We followed the standard rigorous procedure for systematic mapping studies, and complemented with a bottom-up approach to find similar common characteristics of the studies. Finally, we provided different classifications in order to obtain a better understanding of the area.

1.3.2 Exploratory case studies

To address RQ2, we conducted an exploratory study in the form of a multiple case study in industry. The study was designed based on the guidelines for conducting and reporting case studies in software engineering [20]. The process included the assessment of the scenario in four companies through semi-structured interviews with experienced practitioners in the area of heterogeneous computing. In total, ten participants with different roles and from companies in different domains were interviewed face-to-face by two researchers. Each interview had the duration of 30 to 60 minutes and was recorded with the participant's consent. The interview questions were prepared in advance, piloted and discussed among the authors. The data collection followed an interview guide and included the recorded audios and notes taken by the researchers during the interviews. The data analysis was performed using thematic synthesis [20], resulting in a *mindmap* structure (available online [21]). It contains all aspects that were discussed during the interviews organized in clusters.

1.3.3 Design science

To address RQ3, we conducted the studies based on the design science methodology. In summary, design science research (or *constructive science research*) focuses on establishing and operationalizing research when the desired goal is an artifact or a recommendation [22]. The application of this approach results in new ideas or a set of analytical techniques that enable the development of research [23]. In practice, the conducted research activities were separated into 3 stages: *exploration*, *conceptualization* and *evaluation*.

The *exploration* stage consisted of studying the available literature and analyzing the current state-of-the-practice in industrial contexts through workshops and discussions with experts in industry. These meetings occurred in the context of an AI project in partnership with several companies in the Nordics. This stage resulted in the discovery of several concerns and approaches related to the development of software for heterogeneous platforms.

The *conceptualization* stage included an iterative sketch of the solution based on the knowledge acquired during exploration. In this case, we prepared the first versions of the proposed framework and discussed its details with the experts. Several aspects were refined based on their comments through discussions among the researchers involved. Then, we iterated and continued to refine the proposal until it reached a certain level of maturity. Both positive and negative feedback was collected in the meeting occasions. The result was a decision framework containing different stages, aspects, and questions that should be answered.

Then, the *evaluation* stage consisted of preparing, distributing, and analyzing the results of a questionnaire that was sent to several companies. The questionnaire was designed with the intention of capturing the expert's feedback about the framework through explicit, open questions about their daily activities regarding the topic. These questions were derived from the concerns and questions proposed in the framework, allowing for questioning about each stage of the framework. Then, we performed qualitative analysis that resulted in a set of aspects that are described in the publications as the

highlights of the process. In addition to the questionnaire, we also conducted a case study in order to demonstrate the usability of the approach in practice. We selected a functionality in the domain of self-driving vehicles developed within a research facility associated with Chalmers University of Technology.

1.4 Contributions

In this section, we present a summary of the three research contributions, followed by their relation to the papers included in this thesis. These contributions are the outcome of our investigation on the topics that were previously described, aiming to answer the research questions that were elaborated.

The main contributions of this thesis are the following:

1. An overview of the main concerns and approaches of software deployment on heterogeneous platforms;

2. The identification of common practices and challenges when developing software for heterogeneous platforms in industry; and

3. A decision framework for guiding engineers in refactoring software for execution on heterogeneous platforms.

Table 1.1: Mapping between contributions, topics, research questions, papers, and research methods.

Contributions	Topics	Questions	Papers	Research methods
Contribution 1	Software deployment: concerns and approaches	RQ1	Paper A	Systematic mapping study
	Software architecture solutions		Paper B	(literature review)
Contribution 2	Challenges in industry: adoption framework	RQ2	Paper C	Exploratory study (case study)
Contribution 3	Software architecture: principles	RQ3	Paper D	Design science
	Decision framework for migration		Paper E	(case study)

We started by conducting a systematic literature review in the form of a mapping study to obtain an overview of the area through the literature, leading to Contribution 1. This study revealed several concerns that are relevant and approaches that can be used for deploying software on heterogeneous platforms. Then, we conducted an exploratory study that provided an overview of the common practices in industry, including the challenges that companies typically face when developing software for heterogeneous platforms, which led to Contribution 2. Finally, we delivered Contribution 3 by synthesizing the knowledge obtained in the previous studies to propose a systematic approach to guide engineers in the process of migrating software for execution on heterogeneous platforms. The relation between research questions, contributions, papers, and their main topics is shown in Table 1.1. In the next subsections, we provide a summary of the contributions, extracted from the papers.

1.4.1 Contribution 1: An overview of the main concerns and approaches of software deployment on heterogeneous platforms

We systematically searched and analyzed the literature in order to obtain an overview of the research area through the mapping study methodology. We considered papers indexed by trusted libraries in computer science and followed a pre-defined process to formulate the research questions, conduct the research, screen the papers, and extract data from them. Then, the synthesis was performed in the search for gaps, trends, and patterns that allow for an overview of the research area.

Contribution 1 is associated with the following research question:

RQ1: What are the main concerns and approaches in software deployment on heterogeneous platforms?

In the literature, the majority of studies discussing heterogeneous platforms referred to a combination of CPU and GPU processing units, as shown in Figure 1.3.

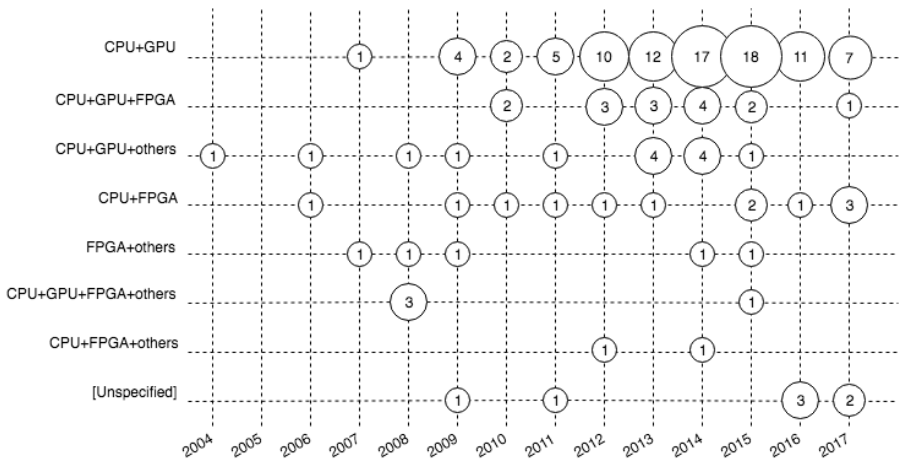


Figure 1.3: Distribution of primary studies according to the types of processors being discussed.

Most primary studies (91.7%) proposed solutions to a particular problem, rather than providing evaluations of existing approaches or opinions on a topic. The validation in the papers mostly focused on demonstrating the applicability of the proposed approach through a simple example or through experiments in a case study.

More than 100 concerns were identified in the selected primary studies. They were organized into the following categories: *scheduling*, *software quality*, *software architecture*, *development process* and *hardware*. Each category contains several sub-categories showing the complexity of the concerns.

Scheduling: refers to the means and the order in which portions of the software are executed, in different levels of granularity. The main discussion in this topic is about load balancing, and how challenging it is to avoid imbalance

in runtime. Scheduling often referred to either tasks, kernels, or complete applications.

Software quality: refers to overall quality attributes of the software, such as performance and maintainability. The most addressed topic is *performance*, followed by *portability*, *efficiency*, *maintainability* and *scalability*.

Software architecture: includes concerns related to the management of the architecture-level attributes of systems containing heterogeneous platforms. The main concern in software architecture is the efficient management of data and memory. This topic includes discussions about the means to transfer data and the communication requirements between processors.

Development process: refers to the activities and challenges that are relevant from the perspective of a software engineering process. The main related concern is the efficiency in the process, including the use of different programming models, vendor-specific tools and libraries. The complexity in developing software for heterogeneous platforms, including parallel programming, is also a concern.

Hardware: includes technical concerns that are directly related to the physical portion of the system. The main concerns are related to the hardware constraints, energy consumption and the possibility of components malfunctioning.

The discovered approaches were classified into *general practices*, *design time practices*, and *runtime practices*. In other words, we clustered practices that can be applied to design time, runtime, or that are applied orthogonally. General practice approaches provide general principles, rather than focusing on life cycle, while theoretical frameworks can be applied to both design and runtime.

General practices: include techniques that are orthogonal to the binding time in the software development process (design time and runtime). These techniques can be used in either or in both stages, such as development frameworks. Most approaches to realize software for heterogeneous platforms are frameworks and APIs, such as CUDA and OpenCL. Further, several load balancing techniques and scheduling algorithms are used to determine which and when each portion of the software is executed on a given processor. In the development process, simulation and visualization techniques are typically used to predict the behavior of the applications.

Design time practices: refer to the set of approaches and methods performed at design time, aiding in the optimization of the mapping between software portions and hardware portions (e.g., *profiling*, cost functions). In several approaches, the tasks and resources are modelled using graphs and abstract specifications, while performance models are used to estimate the execution time of the tasks. In terms of software architecture, the following styles are used: *layered architectures*, *pipelined architectures* and *master-slave architectures*. Some architectural principles are also used, such as: *separation of concerns*, *aspect-oriented architectures*, *standardized architectures*, and the use of *dedicated communication structures*.

Runtime practices: refer to mechanisms that perform changes during runtime, such as running a giving kernel once on all available processors and storing performance results. Several solutions implement their own scheduler that is used at runtime for allocating taks to processors. Further, the schedule

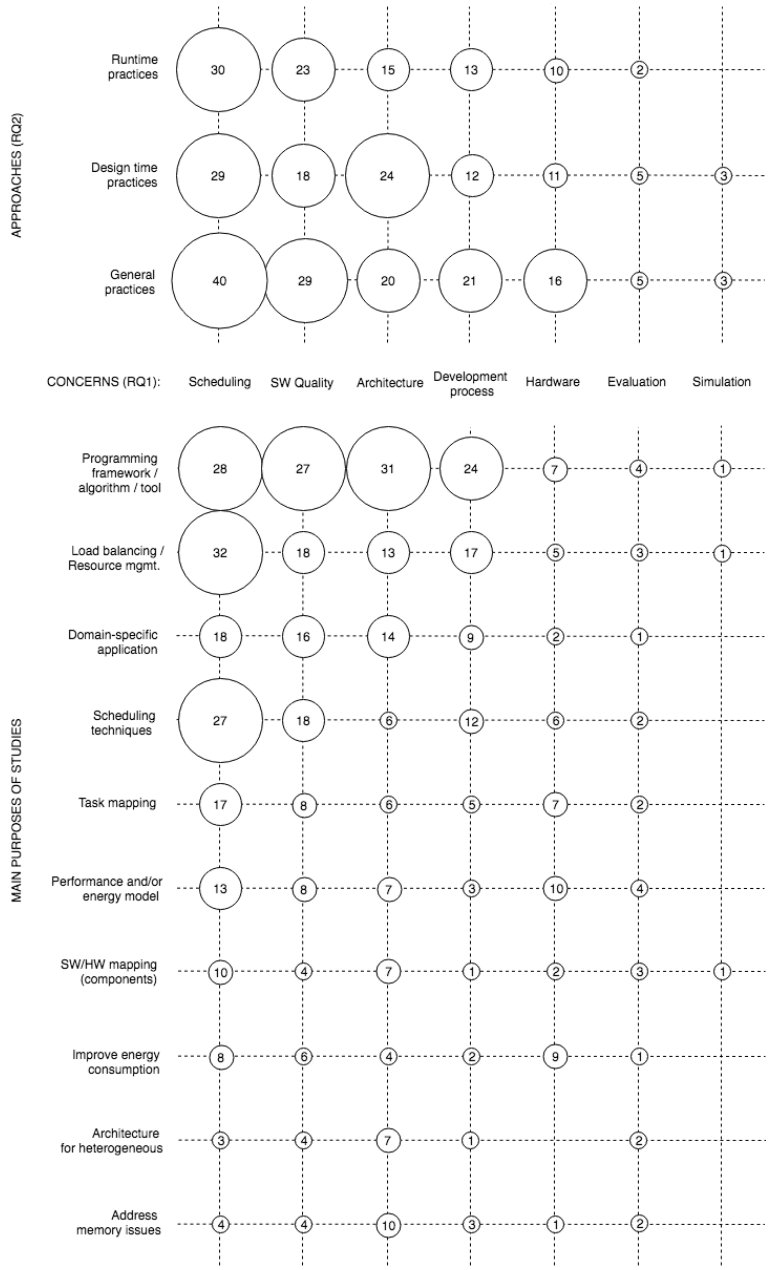


Figure 1.4: Distribution of studies according to the concerns (RQ1), approaches discussed (RQ2) and their main purposes.

at runtime is estimated through *profiling*, which executes small portions of the workloads and their execution times are collected. Other techniques such as *task queueing* and tracking of the states of the jobs are also used.

Figure 1.4 depicts the distribution of studies according to the *concerns*, *approaches* and the main purposes of these studies. It shows a concentration of studies discussing the following concerns: *scheduling*, *software quality*, *architecture*, and slightly less the *development process*. They are almost equally distributed over the approaches i.e., *design time*, *runtime*, and *general practices*. Most studies discussing these concerns have as their main purpose to develop a programming framework, algorithms, or tools for deployment optimization. A large portion of the studies propose practices that are *orthogonal* to the software deployment binding time, and can be applied either at design time or runtime, or represent a more complete approach that includes both. On the other hand, concerns like hardware, evaluation and simulation are not as widely discussed, and thus represent opportunities for research. In summary, we identified a lack of holistic approaches to developing software for heterogeneous platforms, since most studies focused on particular concerns (e.g., scheduling of tasks). Further, there were very few studies that presented validation or evaluation research, which indicates a relative immaturity of the field and thus opportunity for future research.

1.4.2 Contribution 2: An overview of the common practices and challenges when developing software for heterogeneous platforms in industry

We conducted a multiple case study in industry in order to obtain an overview of the common practices and challenges in industry when developing software for heterogeneous platforms. The exploratory study included semi-structured interviews with experts from companies in different domains. The interviews mainly focused on *concerns* and *approaches*, as defined in the previous studies.

Contribution 2 is associated with the following research question:

RQ2: What are the challenges in industry concerning software development for heterogeneous platforms?

The data obtained from the interviews indicated that the companies were in different stages regarding the use of heterogeneous platforms in their products. Thus, the experts in the companies experience scenarios with different characteristics, with different challenges, and use different tactics to address them. To structure the results, we condensed the challenges into a classification that describes the different stages that the companies are in regarding the adoption of heterogeneous platforms. The classification includes 4 stages: *Migrating*, *Establishing a process*, *Scaling up*, and *Refining the process*. In addition to the *challenges* in each stage, we collected the *main goal* of the company in using heterogeneous platforms, the *type of applications* that are developed, the practitioners' take on *software architecture*, and (iv) the type of *toolchains* that are used. The overview of the structure is shown in Figure 1.5.

In the *migrating* stage, the challenges are mostly related to building up hardware-specific expertise in the development teams. Depending on the type

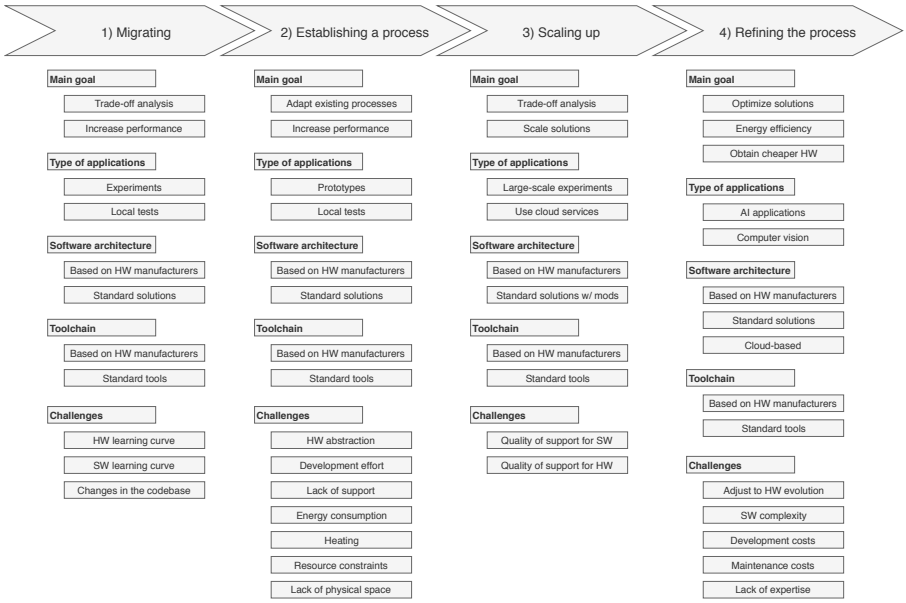


Figure 1.5: Stages of adoption of heterogeneous platforms.

of hardware that is chosen, the learning curve must be taken into account in the trade-off analysis for adoption. When *establishing a process*, there are difficulties in complying with the differences introduced by heterogeneous platforms in the development process. For instance, it is not straightforward to establish a process when new frameworks or deployment routines must be adopted. In the automotive domain, there are challenges related to the introduction of new hardware resources, such as heating, management of energy consumption and lack of physical space. Companies in the *scaling up* stage were concerned with the quality of support that is provided by manufacturers and framework providers. When *refining a process*, there is a challenge in adjusting to hardware evolution, as the product life cycles in some domains are long. Further, there are typically high costs involved in changing the software when new hardware is introduced.

In summary, it is clear that the adoption of heterogeneous platforms requires additional efforts throughout the development process, particularly in the initial phase of using heterogeneous platforms. Practitioners typically rely on the tools and frameworks that are provided by the manufacturers, therefore the topics previously discovered in the literature were either not discussed in the detail or not brought up at all during this study. The concerns identified in the literature are mostly on system level (low-level), rather than on application level. In contrast, the challenges reported by the practitioners are on a higher level of abstraction, such as the lack of hardware expertise and the lack of tool support. Their focus is on the improvement of development processes, organization issues and high-level software design.

1.4.3 Contribution 3: A decision framework for guiding engineers in refactoring software for execution on heterogeneous platforms

Based on the identified challenges in both academia and industry, we proposed a decision framework for migrating software to heterogeneous platforms. The study was performed based on the design science methodology, through which we analyzed the current practices and proposed a solution based on our interactions with experts in industry. In the context of heterogeneous platforms in industry, we identified the need for a decision process that contains all relevant aspects of migration. The decision process must contain steps for engineers to follow in order to reason upon the issues that may arise and effectively prepare the software for execution on heterogeneous platforms. Furthermore, the application of a step-by-step decision process based on predefined steps reduces the risks in the project connected to diverse interpretations and engineering actions.

Contribution 3 is associated with the following research question:

RQ3: What is a relevant process for engineers to follow when migrating software to heterogeneous platforms?

The interactions with experts resulted in initial sketches of the proposed Heterogeneous Platform Migration framework (HPM-Frame), which was evolved through their feedback. It consists of 5 stages, namely: *Assessing*, *Re-architecting*, *Developing*, *Deploying*, and *Evaluating*. Each stage includes a number of main concerns that should be addressed and questions that should be answered in order to obtain decision support for the migration procedure. The stages are briefly described as follows.

Assessing. This stage mainly reasons about the feasibility of system migrating towards heterogeneous platforms. It assesses the advantages, disadvantages, and the preliminary impact of migrating. The bottlenecks of the system must be identified based on a selected set of non-functional properties to be improved. As early as in the assessing stage, the hardware to be used must be identified in order to adjust the subsequent steps accordingly. Then, the possibility of redesigning the identified portion should be preliminarily assessed through the identification of constraints. Further, a preliminary analysis must be conducted in order to determine the impact on the existing software architecture, and the consequences of such changes in the project.

Re-architecting. This stage determines the necessary changes to the software architecture. It focuses on identifying a software/hardware configuration that is suitable according to the predefined non-functional requirements. In other words, it determines which portions of the software will be executed by each processing unit. Initially, it examines the existing data flow, in order to obtain understanding of the present structure and communication behavior. The ultimate goal in this stage is to determine the overall architecture design according to the necessary components, communication requirements, and technology constraints.

Developing. The goal of this stage is to implement the solution through code, realizing the planned changes in the architecture design. It selects a software development process that is suitable according to the context,

including the identification of tools, frameworks and programming languages to be adopted. Furthermore, this stage addresses the problem of refactoring of software components. This practice consists of specifying interfaces between components and rewriting parts of the code to enable their execution on an accelerator.

Deploying. This stage determines the preparations that are for the software to be executed on heterogeneous platforms. It reflects the decisions made in the *re-architecting* stage in order to ensure that the communication requirements are addressed, and the functionalities are preserved. A deployment process is determined, including the frequency of deliveries, timing and resource constraints, and other specific issues. Depending on the project, the need of hardware preparations prior to deployment is also addressed in this stage.

Evaluating. The goal of this stage is to evaluate the solution and the outcomes of the migration process. The main activity in this stage is to ensure that the functionality provided through the migrated software has been preserved. Additionally, the functionalities that were not subject of change must also be tested. Finally, the outcomes of the migration process are evaluated concerning both the expected gains in performance and the project trade-off of undergoing migration.

An overview of HPM-Frame is shown in Table 1.2. It presents the stages, aspects, and questions that are respective to each aspect. The stages follow the reasoning process considering the different phases of software development for heterogeneous platforms. In practice, these phases may not necessarily be followed sequentially as shown in the table, but rather suggest a flow that includes revisiting and feedback loops in the process.

The proposed stages can be integrated with classical software development processes and therefore assume a complementary role to the reasoning that is typically performed in the development applications. Figure 1.6 shows an example of a decision process flow following the decision framework, considering an iterative development process. It highlights the main decision process flow that includes the activities that are preliminary to each aspect being addressed. The feedback loops between the different aspects are highlighted as development iterations, through which relevant aspects that were previously analyzed are revisited.

The evaluation of the framework consisted of the synthesis of data from multiple sources: interviews and questionnaires with industrial experts in the automotive and AI domain (partner companies), and a hybrid academia-industrial environment (case study). We demonstrated the applicability of the approach through a case study in the automotive domain, in which we implemented an edge detector as a 2D convolution operation in two versions: one using CPU only, and the other using a combination of CPU and GPU.

Table 1.2: Overview of the reasoning framework.

Stages	Aspects	Questions
ASSESSING	Functionality analysis	Q1: Are there bottlenecks in the system?
		Q2: Which parts of the system require better performance?
		Q3: Would the selected function perform more efficiently on a different type of processing unit?
		Q4: Is the existing software parallelizable?
	Hardware decision	Q5: What type of hardware should be used?
		Q6: What hardware vendor should be used?
		Q7: What vendor-specific toolchain is needed for the change?
	Impact analysis	Q8: What would be the impact on the existing architecture?
		Q9: What would be the impact on the system's performance and life cycle?
		Q10: What are the expected business benefits (time-to-market, market advantage, user perception)?
	Project feasibility analysis	Q11: What would be the cost of the changes?
		Q12: What would be the costs to maintain and evolve the functionalities once migrated?
		Q13: What would be other consequences of the change regarding non-functional properties?
RE-ARCHITECTING	Software and hardware mapping	Q14: Which functionalities will be executed by the accelerator(s)?
		Q15: How do the potential configurations perform?
		Q16: What will be the mapping between software components and processing units?
	Impact on the software architecture	Q17: What is the impact of the new architecture on the constraints and requirements at run-time?
		Q18: What is the impact of the new architecture on the constraints and requirements related to life cycle and business aspects?
	Overall architecture design	Q19: What will be the new system architecture in terms of new/changed architectural structure, components, and communication?
DEVELOPING	Software development process	Q20: What are the limitations of the existing tools, frameworks, programming languages, hardware?
		Q21: What tools, frameworks and programming languages will be used?
		Q22: What changes will affect the development process (development, testing, cooperative development, etc)?
	Software components refactoring	Q23: What type of interface will be defined for the heterogeneous components?
		Q24: How will data be transferred between the platforms?
DEPLOYING	Deployment process	Q25: Will deployment be available only in a development phase, or will dynamic deployment be available?
		Q26: Which constraints exist in the deployment process (timing constraints, resource capacity, and similar)?
		Q27: Can components be deployed independently, or all components must be (re)deployed as a set?
		Q28: What are the capabilities of saving data and system status on the target processing units?
	Hardware preparation	Q29: What are the preparations needed for execution?
EVALUATING	Functionality testing	Q30: Have the functionalities been preserved despite refactoring?
	Refactoring outcomes	Q31: Were the expected gains in performance achieved (e.g., execution time, maintainability)?
		Q32: What was the cost or refactoring?
		Q33: What was the trade-off (cost/effort) of refactoring vs. the gains in performance?

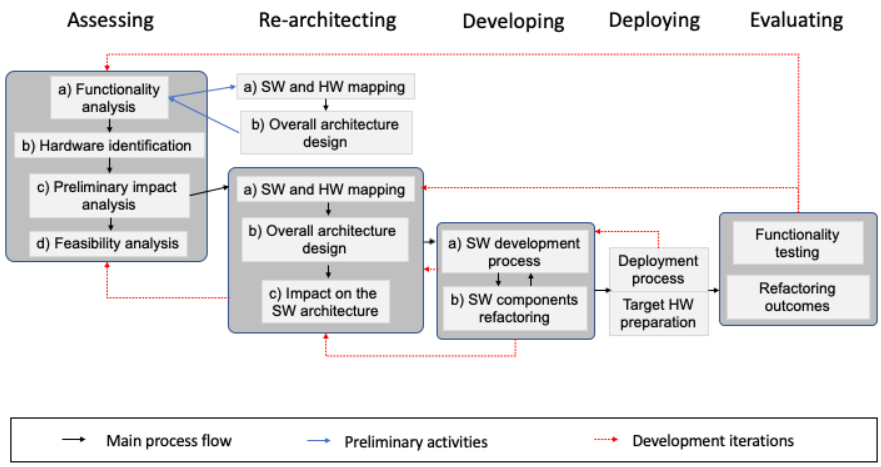


Figure 1.6: Decision process flow within the framework.

1.5 Publications

In this section, we list the publications appended to this thesis. The complete version of the papers can be found in the chapters following this introduction. They have been reformatted in order to comply with the layout of this thesis.

Paper A

Software Deployment on Heterogeneous Platforms: A Systematic Mapping Study

H. Andrade, J. Schröder, I. Crnkovic

IEEE Transactions on Software Engineering journal (TSE)

Abstract: *Context:* Multiple types of processing units (e.g., CPUs, GPUs and FPGAs) can be used jointly to achieve better performance in computational systems. However, these units are built with fundamentally different characteristics and demand attention especially towards software deployment. *Objective:* The goal of this work is to summarize the state–of–the–art of software deployment on heterogeneous platforms. We provide an overview of the research area by searching for and categorizing relevant studies, as well as discussing gaps and trends of the field. We are interested in the main concerns (RQ1) and the approaches used (RQ2) when deploying software on heterogeneous platforms. *Method:* In order to achieve our goal, we performed a systematic mapping study, which refers to a method for reviewing literature with basis on predefined search strategies and a multi–step selection process. *Results:* We selected and analyzed 146 primary studies from multiple sources and found that the area of research is dominated by solution proposals. The

majority of the studies discuss concerns about scheduling, the quality of the software, and its architecture. A large number of studies focuses on the problem of scheduling tasks and processes. We found approaches that are applied at different binding times (i.e., design time, runtime, orthogonal). *Conclusion:* The evaluation of the proposed solutions in an industrial context is missing. Also, the proposed methods have not been evaluated in development processes. Most of the methods address a particular concern, or a few concerns, while there is a lack of a holistic approach.

Paper B

A Review on Software Architectures for Heterogeneous Platforms
H. Andrade, I. Crnkovic

Proceedings of the 25th Asia-Pacific Software Engineering Conference (APSEC). Nara, Japan, 4-7 December, 2018

Abstract: The increasing demands for computing performance have been a reality regardless of the requirements for smaller and more energy efficient devices. Throughout the years, the strategy adopted by industry was to increase the robustness of a single processor by increasing its clock frequency and mounting more transistors so more calculations could be executed. However, it is known that the physical limits of such processors are being reached, and one way to fulfill such increasing computing demands has been to adopt a strategy based on heterogeneous computing, i.e., using a heterogeneous platform containing more than one type of processor. This way, different types of tasks can be executed by processors that are specialized in them. Heterogeneous computing, however, poses a number of challenges to software engineering, especially in the architecture and deployment phases. In this paper, we conduct an empirical study that aims at discovering the state-of-the-art in software architecture for heterogeneous computing, with focus on deployment. We conduct a systematic mapping study that retrieved 28 studies, which were critically assessed to obtain an overview of the research field. We identified gaps and trends that can be used by both researchers and practitioners as guides to further investigate the topic.

Paper C

Software Challenges in Heterogeneous Computing: A Multiple Case Study in Industry
H. Andrade, L. Lwakatare, I. Crnkovic, J. Bosch

Proceedings of the 45th IEEE Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA). Kallithea-Chalkidiki, Greece, 28-30 August, 2019

Abstract: One way to improve the performance of embedded systems is through heterogeneous platforms, i.e., using hardware containing more than one type of processor, like CPU + GPU or CPU + FPGA. This approach has shown improved performance, particularly in the domain of artificial intelligence, in which computationally demanding models must be trained and executed. However, these computational environments pose various challenges to software engineering, since applications must be designed and developed differently while accounting for target hardware architectures that are inherently different. Companies interested in migrating to heterogeneous platforms must be aware of the changes to the software development processes that are required to accommodate such solution. In this paper, we conducted a multiple case study that aims to discover the challenges and common practices when developing software for heterogeneous platforms in industrial contexts. First, we organized semi-structured interviews with companies on the automotive, automation, and telecommunication domains. Then, we analyzed and structured the data in order to create a classification describing the software engineering challenges faced by companies using heterogeneous platforms. The companies involved in this study are in different stages of maturity concerning the use of heterogeneous platforms. Thus, the classification takes into consideration these levels to describe the challenges accordingly.

Paper D

Principles for Re-architecting Software for Heterogeneous Platforms
H. Andrade, C. Berger, I. Crnkovic, J. Bosch

Proceedings of the 27th Asia-Pacific Software Engineering Conference (APSEC). Singapore, 1-4 December, 2020

Abstract: The demands on software continues to increase through the constant addition of functionalities and high expectations from users. In particular, *performance* has been the focus in many projects with the goal of fulfilling complex and hard requirements across a variety of domains. One way to achieve satisfactory levels of performance is through heterogeneous computing, i.e., systems that contain more than one type of processing unit, such as CPUs, GPUs, and FPGAs. However, applications are typically designed to be executed on CPUs, and re-architecting software for execution on such heterogeneous hardware architectures entails several challenges that must be addressed. In this paper, we propose a framework that supports engineers in the process of making architectural decisions to re-architect software for execution on heterogeneous platforms. We present several relevant aspects that should be addressed in the process, along with suggestions on how to create design solutions using different existing approaches. The framework was developed based on multiple interactions with three industrial partners and evaluated through a computer vision application in the automotive domain.

Paper E

HPM-Frame: A Decision Framework for Executing Software on Heterogeneous Platforms

H. Andrade, O. Benderius, C. Berger, I. Crnkovic, J. Bosch

Journal of Systems and Software (JSS), 2020 [Submitted]

Abstract: Heterogeneous computing is one of the most important computational solutions to meet rapidly increasing demands on system performance. It typically allows the main flow of applications to be executed on a CPU while the most computationally intensive tasks are assigned to one or more accelerators, such as GPUs and FPGAs. The refactoring of systems for execution on such platforms is highly desired but also difficult to perform, mainly due the inherent increase in software complexity. After exploration, we have identified a current need for a systematic approach that supports engineers in the refactoring process—from CPU-centric applications to software that is executed on heterogeneous platforms. In this paper, we introduce a decision framework that assists engineers in the task of refactoring software to incorporate heterogeneous platforms. It covers the software engineering life cycle through five steps, consisting of questions to be answered in order to successfully address aspects that are relevant for the refactoring procedure. We evaluate the feasibility of the framework in two ways. First, we capture the practitioner's impressions, concerns and suggestions through a questionnaire. Then, we conduct a case study showing the step-by-step application of the framework using a computer vision application in the automotive domain.

1.6 Threats to Validity

In this section, we provide an overview of the threats to the validity of this work. They are discussed using a classification introduced in [24], which distinguishes validity between the following aspects: *construct* validity, *internal* validity, *external* validity, and *reliability*. Detailed threats to validity to each included paper are discussed in their respective chapters.

Construct validity addresses the extent to which operational measures that are studied actually represent the researchers' intentions according to the research questions [25]. The findings in Paper A and Paper B could have potentially been affected by this threat, since the data extraction process was extensive and involved more than one researcher. In order to mitigate this risk, we held weekly meetings between the researchers to align concepts, obtain a common understanding of the topic, and create the research questions. Constant meetings also enabled a systematic process for applying the inclusion/exclusion criteria in the paper selection procedure. The results of each iteration were discussed in meetings and disagreements were solved in these occasions by a third researcher. In Paper C, the threats to construct validity include possible diverse interpretations by the involved parties: researchers, and experts in industry. We mitigated these threats through discussing and reviewing an

interview guide among the researchers, verifying that the interview questions were aligned with the aim of the study. The interview guide was piloted prior to the conduction of the interviews. Further, common terminology and concepts were established in the beginning of each interview in order to minimize the risk of misunderstandings. Such strategy to minimize diverse understanding among stakeholders was also used in Paper D and Paper E, in which we structured the questionnaire, interviews, discussions, and case studies with focus on the proposed framework. We explicitly addressed each stage in the framework in order to reduce the risk of mismatching in the evaluation.

Internal validity concerns the examination of the causal relations in a study [25]. It refers to situations in which the researcher establishes a causal relationship between two events without considering the possibility of additional factors. In Papers A and B, there is a risk that the conclusions about the state-of-the-art drawn from the primary studies were also affected by papers outside the selected set. In order to minimize this risk, we conducted a standard systematic mapping study method that includes a predefined set of inclusion/exclusion criteria, pilot searches, and the calibration of the search string. We reduced the risk of missing relevant studies by performing the search in leading academia databases in the field. In Paper C, we discussed the challenges faced by the experts in the area from their own perspective. There could exist other external factors in industrial contexts leading to those challenges that we were not aware of. We minimized this risk by interviewing experts with different experience levels and from different companies. Regarding Papers D and E, the research activities were designed with basis on the findings from our previous studies and discussions we had with experts in the area. There is a possibility that external factors were not considered in understanding the context, therefore leading to misleading interpretations of the phenomena. To minimize this risk, we structured the data and elaborated a set of steps that were evaluated both qualitatively (via interviews and a questionnaire) and quantitatively (via a case study) by experts in multiple companies.

External validity concerns the extent to which it is possible to generalize the findings, and to what extent the findings are of interest to people outside the investigated case [25]. This threat is potentially present in this thesis mainly in Papers C, D and E. The findings in Paper C concern specific scenarios in a limited set of companies. Generalizing such findings may be a challenge, since the processes established in other companies might differ to a great extent. However, we minimize this threat by standardizing the data analysis and presenting the findings in the form of a model. In different contexts, this model can be used as basis to understanding cases that have similar characteristics of those investigated in this study. Regarding Papers D and E, there is an assumption that the questions would still be valid in different contexts. The proposal in the framework is based on the experiences shared by the participating experts, bringing a narrow perspective of the phenomenon in a particular context of their work. The scenarios can vary greatly in different companies, thus generalizing the results may be a challenge. Assessing the effects of applying the framework in such different contexts is part of future work. We minimized this threat by including multiple practitioners with different levels of experience in the procedures. The application of the framework was restricted to an example in the automotive domain.

Reliability refers to the extent to which the data and analysis are dependent on the specific researchers [25]. In this thesis, we aimed at maintaining the reproducibility of the studies by using well-established, systematic approaches for research. In particular in Papers A and B, we systematically reviewed the literature, establishing inclusion/exclusion criteria beforehand, and elaborating a review protocol. In Paper C, we elaborated an interview guide that was used when interacting with all the companies involved in the study. In Paper C, the questions in the questionnaire were created with focus on clarity and the interviews allowed experts to request face-to-face clarification when needed. The interviews were conducted based on a preproduced interview guide. In Papers D and E, we proposed a systematic, step-by-step approach to migrating software to heterogeneous platforms in order to reduce the possibility of diverse interpretations. The case studies were conducted by following the steps of the framework in the scenarios presented in the cases. All assets produced in the conduction of the studies have been made publicly available for verification and reproducibility.

1.7 Conclusion

Heterogeneous computing is a technology that enables higher levels of performance in computer systems by employing hardware platforms containing more than one type of processor. This technology allows for lower cost and energy usage compared to homogeneous, CPU-based platforms, since different types of data can be processed by units that are specialized in them. Heterogeneous computing has been gaining importance lately particularly due to the high demands of performance in, among others, artificial intelligence systems, which include machine learning and deep learning applications. In addition to lower execution time through software/hardware mapping, heterogeneous platforms may also achieve lower levels of energy consumption, which are of primary importance in domains such as embedded systems. The current state of practice indicates that it typically is challenging to obtain expertise in the area, as well as tools and processes that cover the entire software engineering process. Furthermore, most applications are built initially for CPUs and then ported manually to heterogeneous platforms, suggesting a strong need in industry for an approach that supports engineers (developers, analysts, architects) in the process of migrating software for execution on heterogeneous platforms.

This thesis presents the motivation, procedures, and findings of our research conducted in the area of software deployment on heterogeneous platforms. The overall goal of the thesis is to provide engineers with a decision framework for migrating software for execution on heterogeneous platforms. We investigated both academia and industrial contexts in order to provide: (i) an overview of the state-of-the-art of software deployment on heterogeneous platforms; (ii) an overview of the common challenges and practices in industry when developing software for heterogeneous platforms, and (iii) a decision framework that supports reasoning in the task of migrating software for execution on heterogeneous platforms.

We provided an overview of the state-of-the-art through an extensive systematic literature study that “mapped out” the field of research, obtaining

domain knowledge, and identifying gaps that indicated possibilities for future investigation. Several concerns and approaches for software deployment on heterogeneous platforms were identified and discussed. Most concerns in the area are related to particular problems in implementing such systems, such as the scheduling of tasks and processes, ensuring software quality, and designing a software architecture that is most suitable. In terms of existing approaches, the majority of proposals are directed to general practices, i.e., practices that can be applied to either design time or runtime, or that are applied orthogonally. The majority of the primary studies refer to solution proposals, with few evaluation or validation studies in industrial contexts, which represents opportunity for future research in the area. Further, the study showed that a number of existing architectural design principles and architectural patterns can be used in the development of software for heterogeneous platforms.

In terms of challenges and practices in industry, we found that companies that are in different stages of adoption of heterogeneous platforms typically face different challenges regarding to (i) actually migrating existing software; (ii) establishing a process for developing software for such platforms; (iii) scaling up production in software development; and (iv) refining the software engineering process to optimize results. On several occasions, the companies mentioned difficulties in obtaining expertise in the area, as well as the lack of existing tools and processes that cover the complete development life cycle. Other challenges were also mentioned, such as the difficulties in managing software complexity and establishing continuous deployment & integration of these systems.

Given the results of the aforementioned studies, we proposed a decision framework to guide engineers in the process of migrating software to heterogeneous platforms. The framework contains 5 stages that should be followed, namely: *Assessing*, *Re-architecting*, *Developing*, *Deploying* and *Evaluating*. The proposed aspects are connected to a typical software engineering process that allows for feedback loops and re-visitation of other stages. Each stage includes aspects that should be observed and questions that should be answered in order to obtain reasoning in the migration process. The details were elaborated with basis on the perspectives we obtained from both academia and industry. The framework was presented to – and evolved with – multiple industrial partners in the form of meetings and workshops. The relevance of the approach was evaluated through a set of interviews and a questionnaire that was sent out in order to obtain feedback from these experts. The applicability of the approach was demonstrated through a case study in the automotive domain. We have observed that good practice in the adoption of heterogeneous computing typically consists of incremental engineering, including experimentation, simulation, and continuous integration and deployment of parts of the system.

1.8 Future Work

As future work, we intend to continue investigating topics in the area of software engineering for heterogeneous platforms by considering the following possibilities:

Performing evaluation studies. As discovered in the mapping study, there are very few studies evaluating existing techniques, while the vast majority of

them proposes solutions to a given problem. Our main intention in the future is to evaluate the proposed framework in an industrial scenario. The goal of this next study is to observe how effective the use of the framework is when a well defined software engineering process is already in place. For instance, the focus would be on the measurable impact of the adoption of the framework through experiments. One possibility is to organize focus groups and discuss the modifications in the processes among different stakeholders. We intend to observe the differences in companies with different levels of maturity in adopting heterogeneous platforms.

Further exploring industrial settings. Especially with the popularization of artificial intelligence techniques, heterogeneous platforms can help in improving the performance of software systems in industry. It would be relevant to investigate how heterogeneous platforms can be useful in the context of modern, typically industrial software development processes, such as DevOps. Further, one could also investigate the integration between our proposed framework with existing well established software development frameworks for heterogeneous platforms, such as OpenCL.

Other domains & compliance. The case studies we conducted focused primarily on automotive systems. We intend to apply our proposed approach in different domains in order to capture differences and similarities in the results we obtained from such domain. The proposal then can be extended into either an umbrella framework that is applicable to systems regardless of their domains, or the framework could be further refined to incorporate domain-specific characteristics. For instance, in the case of automotive systems, the stages in framework could comply with standards that are widely used today, such as Simulink models, ISO 26262, and/or other regulatory norms that are currently in effect.